Getting Started with Hugo

From Installation to Your First Professional Website

Prerequisites Check

Required Software:

```
# Check if you have Git (required)
git --version

# We'll install Hugo together
# No other dependencies needed!
```

Optional but Helpful:

- Text editor (VS Code Recommended)
- Basic knowledge of Markdown
- Command line familiarity

What We'll Create:

1. **Personal portfolio site** showcasing your projects
2. **API documentation site** for your Laravel API
3. **Blog setup** for technical writing

Step 1: Install Hugo (Skip if you already installed Hugo)

macOS Installation:

```
# Option 1: Using Homebrew (recommended)
brew install hugo

# Option 2: Using MacPorts
sudo port install hugo

# Verify installation
hugo version
```

Windows Installation:

```
# Option 1: Using Chocolatey
choco install hugo-extended

# Option 2: Using Scoop
scoop install hugo-extended

# Option 3: Download binary from GitHub
# https://github.com/gohugoio/hugo/releases

# Verify installation
hugo version
```

Linux Installation:

```
# Ubuntu/Debian
sudo apt install hugo

# Fedora
sudo dnf install hugo

# Arch Linux
```

Step 2: Create Your First Hugo Site

Initialize New Site:

```
# Create a new Hugo site
hugo new site my-portfolio

# Navigate to the directory
cd my-portfolio

# Initialize Git repository
git init

# See what Hugo created
ls -la
```

Hugo Site Structure:

```
my-portfolio/
├── archetypes/          # Content templates
│   └── default.md
├── assets/              # Files to be processed
├── content/             # Your content (Markdown files)
├── data/                # Configuration files
├── layouts/             # HTML templates
├── static/              # Static files (images, CSS, JS)
├── themes/              # Downloaded themes
├── hugo.toml            # Main configuration file
└── public/              # Generated site (after hugo build)
```

Important Note:

The `content/` folder is where you'll spend most of your time writing!

Step 3: Choose and Install a Theme

Browse Available Themes:

- Visit: https://themes.gohugo.io/
- Filter by: Portfolio, Documentation, Blog
- Popular choices: Ananke, Docsy, PaperMod, Academic

Theme Installation:

Option 1: Git clone (recommended)

1. Change to your Hugo project
2. Run `git clone` to clone any themes you choose.

```
git clone https://github.com/theNewDynamic/gohugo-theme-ananke.git themes/ananke
```

or

```
git clone https://github.com/google/docsy themes/docsy
```

Important: you must delete the .git and .github directories in the theme directory before pushing to GitHub to avoid errors in GitHub.io

Option 2: Download and extract (no Git)

```
curl -L https://github.com/theNewDynamic/gohugo-theme-ananke/archive/master.zip -o ananke.zip
unzip ananke.zip -d themes/
mv themes/gohugo-theme-ananke-master themes/ananke
```

- If you clone directly, it is simpler, but no updates are possible.
- If you want to update themes automatically, use `git submodule`.

(Optional) Install a Theme (Example: Ananke):

```
# Add theme as Git submodule
git submodule add https://github.com/theNewDynamic/gohugo-theme-ananke.git themes/ananke

# Update configuration to use theme
echo "theme = 'ananke'" >> hugo.toml
```

Step 4: Configure Your Site

Edit `hugo.toml` Configuration:

We will use GitHub.io to publish your project.

```
baseURL = 'https://your-username.github.io'
languageCode = 'en-us'
title = 'John Doe - Software Developer'
theme = 'ananke'

[params]
  author = "John Doe"
  description = "Full-stack developer specializing in Laravel and modern web technologies"
  github = "https://github.com/yourusername"
  linkedin = "https://linkedin.com/in/yourusername"
  email = "john@example.com"

[menu]
  [[menu.main]]
    name = "Home"
    url = "/"
    weight = 1
  [[menu.main]]
    name = "Projects"
    url = "/projects/"
    weight = 2
  [[menu.main]]
    name = "Blog"
```

Step 5: Create Your First Content

Create an About Page:

```
# Generate about page
hugo new about.md

# Edit the file
```

Edit `content/about.md`:

Important!

Make sure that `draft = false` to show your contents.

```
---
title: "About Me"
date: 2024-01-15
draft: false
---

# About John Doe

I'm a passionate software developer with expertise in:

...

## Contact
- Email: john@example.com
- GitHub: [github.com/yourusername](https://github.com/yourusername)
- LinkedIn: [linkedin.com/in/yourusername](https://linkedin.com/in/yourusername)
```

Step 6: Create Project Showcase

Create Projects Section:

```
# Create projects directory and first project
hugo new projects/_index.md
hugo new projects/student-api.md
```

You can copy the files in the `code/5_Hugo/content` directory for tests, but you **must** submit your own API for project results.

Edit `content/projects/_index.md`:

```
---
title: "My Projects"
date: 2024-01-15
draft: false
---

# My Development Projects

Here are some of the projects I've built during my software engineering journey.

Each project demonstrates different technical skills and problem-solving approaches.
```

Edit `content/projects/student-api.md`:

```
...

## API Endpoints

### Students Management
| Method | Endpoint                | Description                |
|--------|-------------------------|----------------------------|
| GET    | `/api/students`         | Retrieve all students      |
| GET    | `/api/students/{id}`    | Retrieve specific student  |
| POST   | `/api/students`         | Create new student         |
| PUT    | `/api/students/{id}`    | Update student             |
| DELETE | `/api/students/{id}`    | Delete student             |
```

Step 7: Start Development Server

Launch Hugo Development Server:

```
# Start server with draft content
hugo server -D

# Open your browser to:
# http://localhost:1313
```

What You'll See:

- **Live reloading**: Changes appear instantly
- **Navigation menu**: Home, Projects, Blog, About
- **Professional design**: Thanks to the theme
- **Mobile responsive**: Works on all devices

Development Features:

- Hot reload on file changes
- Draft content preview
- Fast build times (<1 second)
- Real-time error reporting

Step 8: Add Blog Functionality

Create Your First Blog Post:

```
# Create a new blog post
hugo new posts/getting-started-with-hugo.md
```

Edit `content/posts/getting-started-with-hugo.md`:

```
---
title: "Getting Started with Hugo Static Site Generator"
date: 2024-01-15
draft: false
tags: ["Hugo", "Static Sites", "Web Development"]
categories: ["Tutorial"]
---
...
```

Step 9: Customize Your Theme

Basic Theme Customization:

Custom CSS (Optional):

```
# Create custom CSS file
mkdir -p static/css
touch static/css/custom.css
```

Add to `static/css/custom.css`:

```
/* Custom styles for your site */
:root {
  --primary-color: #3498db;
  --secondary-color: #2c3e50;
  --accent-color: #e74c3c;
}
```

Step 10: Build and Preview

Build for Production:

```
# Generate static files
hugo

# Check the output
ls public/
```

The `public/` directory has all the static files (CSS, HTML, and many others) that are deployed.

What Gets Generated:

```
public/
├── index.html              # Homepage
├── about/
│   └── index.html          # About page
├── projects/
│   ├── index.html          # Projects listing
│   └── student-api/
│       └── index.html      # Project detail
├── posts/
│   ├── index.html          # Blog listing
│   └── getting-started-with-hugo/
│       └── index.html      # Blog post
├── css/                    # Stylesheets
├── js/                     # JavaScript
└── images/                 # Optimized images
```

Test Production Build:

```
# Serve the built site locally
cd public && python -m http.server 8080
# Visit: http://localhost:8080
```

Step 11: Content Organization Best Practices

Recommended Content Structure (Example):

```
content/
├── _index.md                  # Homepage content
├── about.md                   # About page
├── projects/
│   ├── _index.md              # Projects listing page
│   ├── student-api.md         # Laravel API project
│   ├── docker-setup.md        # Docker learning project
│   └── portfolio-site.md      # This Hugo site
├── posts/                     # Blog posts
│   ├── _index.md              # Blog homepage
│   ├── hugo-tutorial.md       # Technical tutorials
│   └── docker-guide.md        # Learning experiences
└── docs/                      # Documentation
    ├── _index.md              # Documentation home
    ├── api/
    │   ├── _index.md          # API docs overview
    │   └── authentication.md
    └── setup/
        ├── installation.md
```

The _index.md file

- When you don't have the _index.md: Hugo automatically generates a list page (behind the scenes) showing all content in that section using your theme's default template.
- With _index.md: You take manual control over what appears on that index page.

Make _index.md so you can control what is displayed.

Step 12: Add Images and Media

Create image-test.md for testing.

```
my-portfolio> hugo new image-test.md
Content "/Users/smcho/temp/hugo/my-portfolio/content/image-test.md" created
```

Adding Images:

```
# Create images directory
mkdir -p static/images

# Add project screenshots
# Files are already copied in the code directory
cp ~/screenshots/student-api.png static/images/
cp ~/screenshots/docker-dashboard.png static/images/
```

Reference Images in Content:

```
## Project Screenshot

![Student API Dashboard](/images/student-api.png)

## Architecture Diagram

![Docker Architecture](/images/docker-dashboard.png)
```

Notice the leading `/` to access the images.

Optimize Images:

```
# Hugo can automatically optimize images
# Place in assets/ instead of static/
mkdir -p assets/images
## copy image (student-api2.md) in this directory
```

How Hugo Image Processing Works

1. Location Matters:

- `static/images/` → Served as-is (no processing)
- `assets/images/` → Available for processing (but not automatic)

2. You Need to Trigger Processing:

```
<!-- This will be optimized -->
<!-- These functions tell Hugo to look in assets/ -->
{{< figure src="images/student-api2.png" width="300" >}}

<!-- This does NOT process (just displays original) -->
<!-- These paths tell Hugo to look in static/ -->
![Student API](/images/student-api2.png)
```

Step 13: Advanced Configuration

Enhanced `hugo.toml`:

```
baseURL = 'https://yourusername.github.io'
languageCode = 'en-us'
title = 'John Doe - Full Stack Developer'
theme = 'ananke'

# Enable emoji support
enableEmoji = true

# Enable syntax highlighting
[markup]
  [markup.goldmark]
    [markup.goldmark.renderer]
      unsafe = true
  [markup.highlight]
    style = 'github'
    lineNos = true
```

Navigation menu

```toml
[menu]
  [[menu.main]]
    name = "Home"
    url = "/"
    weight = 1
  [[menu.main]]
    name = "Projects"
    url = "/projects/"
    weight = 2
  [[menu.main]]
    name = "Blog"
    url = "/posts/"
    weight = 3
  [[menu.main]]
    name = "Documentation"
    url = "/docs/"
    weight = 4
  [[menu.main]]
    name = "About"
    url = "/about/"
    weight = 5
```

Taxonomies (tags and categories)

```toml
[taxonomies]
  tag = "tags"
  category = "categories"
  technology = "technologies"
```

With your markdown headers.

```
---
title: "E-commerce Web App"
categories: ["programming"]
tags: ["api", "python"]
technologies: ["react", "express", "mysql", "flask"]
---
```

Hugo will generate these pages.

```
/tags/python/           ← All posts tagged "python"
/tags/api/              ← All posts tagged "api"
/categories/programming/ ← All programming posts
/technologies/flask/    ← All Flask-related posts
```

Output format

How Hugo Categorizes Your Content:

```
content/
├── about.md                ← "page" type
├── posts/                  ← "section" type
│   ├── post1.md          ← "page" type (inside section)
│   └── post2.md          ← "page" type (inside section)
└── projects/               ← "section" type
    └── project1.md       ← "page" type (inside section)
```

Individual Files = "page"

```
page = ["HTML"]
```

Applies to: Any single .md file

- content/about.md → /about/index.html
- content/posts/my-post.md → /posts/my-post/index.html

Directories with Multiple Files = "section"

```
section = ["HTML", "RSS"]
```

Applies to: Any directory containing multiple content files

- content/posts/ → /posts/index.html + /posts/index.xml
- content/projects/ → /projects/index.html + /projects/index.xml

our site's root page (/) = "home"

```
# Output formats
[outputs]
  home = ["HTML", "RSS", "JSON"]
```

Generates:

```
public/
├── index.html     ← "home" HTML output
├── index.xml      ← "home" RSS output
└── index.json     ← "home" JSON output
```

Access URLs:

```
http://localhost:1313/          ← index.html (homepage)
http://localhost:1313/index.xml ← RSS feed
http://localhost:1313/index.json ← JSON API
```

Your File Structure:

```
content/
├── about.md
└── posts/
    ├── post1.md
    └── post2.md
```

Hugo Generates:

```
public/
├── about/
│   └── index.html          ← "page" format
├── posts/
│   ├── index.html          ← "section" format
│   ├── index.xml           ← "section" format (RSS)
│   ├── post1/
│   │   └── index.html      ← "page" format
│   └── post2/
│       └── index.html      ← "page" format
```

Step 14: SEO and Performance Optimization

Add SEO Meta Tags:

Create `layouts/partials/head-additions.html`:

```
<!-- Custom meta tags -->
<meta name="description" content="{{ .Description | default .Site.Params.description }}">
<meta name="keywords" content="{{ .Site.Params.keywords | delimit ", " }}">
<meta name="author" content="{{ .Site.Params.author }}">

...

<!-- Favicon -->
<link rel="icon" type="image/x-icon" href="/favicon.ico">
<link rel="apple-touch-icon" sizes="180x180" href="/apple-touch-icon.png">
```

Performance Optimization:

```
# In hugo.toml
[minify]
  disableCSS = false
  disableHTML = false
  disableJS = false
  disableJSON = false
  disableSVG = false
  disableXML = false

[imaging]
  quality = 85
  resampleFilter = "lanczos"
```

Step 15: Content Templates (Archetypes)

What are Archetypes?

Archetypes are **content templates** that Hugo uses when you create new content with `hugo new`. They help ensure consistency across your site by providing:

- Standard front matter structure
- Default content layout
- Predefined metadata fields
- Time-saving automation

Understanding the Default Archetype:

Every Hugo site starts with `archetypes/default.md`:

```
---
title: "{{ replace .Name "-" " " | title }}"
date: {{ .Date }}
draft: true
---
```

Template Variables Explained:

- `{{ .Name }}`: The filename you specify
- `{{ .Date }}`: Current date/time
- `{{ replace .Name "-" " " | title }}`: Converts "my-post" → "My Post"

Enhanced Default Archetype:

Edit `archetypes/default.md` for better content structure:

```
---
title: "{{ replace .Name "-" " " | title }}"
date: {{ .Date }}
draft: true
tags: []
categories: []
description: ""
author: "{{ .Site.Params.author | default "Your Name" }}"
---

# {{ replace .Name "-" " " | title }}
Brief introduction to the topic.
...
## Conclusion
Summarize the main points and next steps.
```

Creating Custom Archetypes:

Blog Post Archetype:

Create `archetypes/posts.md` :

```
---
title: "{{ replace .Name "-" " " | title }}"
date: {{ .Date }}
draft: true
tags: ["tutorial", "web-development"]
categories: ["programming"]
description: "{{ replace .Name "-" " " | title }} - A comprehensive guide"
author: "{{ .Site.Params.author }}"
readingTime: "5 min"
---

...
```

Project Showcase Archetype:

Create `archetypes/projects.md` :

```
---
title: "{{ replace .Name "-" " " | title }}"
date: {{ .Date }}
draft: true
tags: []
categories: ["project"]
technologies: ["HTML", "CSS", "JavaScript"]
projectUrl: ""
githubUrl: ""
demo: ""
status: "In Progress" # Completed, In Progress, Planning
difficulty: "Intermediate" # Beginner, Intermediate, Advanced
---


...
```

API Documentation Archetype:

Create `archetypes/docs.md`:

```
---
title: "{{ replace .Name "-" " " | title }}"
date: {{ .Date }}
draft: true
categories: ["documentation"]
tags: ["api", "reference"]
api_version: "v1.0"
weight: 10
---

...
```

Using Custom Archetypes:

```
# Use default archetype
hugo new about.md

# Use posts archetype
hugo new posts/my-tutorial.md

# Use project archetype
hugo new projects/student-api.md

# Use docs archetype
hugo new docs/api/authentication.md
```

In this example, Hugo generates pages in subdirectories using the templates in the corresponding markdown files.

How Hugo Chooses Archetypes:

1. If creating `posts/article.md` → looks for `archetypes/posts.md`
2. If `posts.md` doesn't exist → uses `archetypes/default.md`
3. Section name must match archetype filename

Advanced Archetype Features:

Conditional Content:

```
---
title: "{{ replace .Name "-" " " | title }}"
date: {{ .Date }}
{{ if eq .Section "projects" }}
technologies: []
projectUrl: ""
{{ else if eq .Section "posts" }}
tags: ["tutorial"]
readingTime: "5 min"
{{ end }}
---

{{ if eq .Section "projects" }}
## Project Overview
{{ else }}
## Introduction
{{ end }}
```

Testing Your Archetypes:

```
# Create test content
hugo new posts/test-post.md
hugo new projects/test-project.md
hugo new docs/test-doc.md

# Check the generated files
cat content/posts/test-post.md
cat content/projects/test-project.md
cat content/docs/test-doc.md
```

Common Commands Reference

Daily Hugo Workflow:

```
# Development
hugo server -D               # Start dev server with drafts
hugo server --bind 0.0.0.0   # Access from other devices
hugo server --port 8080      # Use different port

# Content creation
hugo new posts/new-post.md          # New blog post
hugo new projects/new-project.md    # New project
hugo new docs/api/endpoint.md       # New documentation

# Building
hugo                         # Build for production
hugo --minify                # Build with minification
hugo --baseURL="https://example.com"  # Override base URL

# Cleanup
hugo mod clean               # Clean module cache
```

Useful Flags:

- `-D` : Include draft content
- `--minify` : Minify output
- `--gc` : Run garbage collection
- `--verbose` : Show detailed output
- `--cleanDestinationDir` : Clean destination before building

Troubleshooting Common Issues

Issue 1: Theme Not Working

```
# Check the theme is properly installed
ls themes/

# Verify theme name in config
grep theme hugo.toml

# Ensure theme is compatible with Hugo version
hugo version
```

Issue 2: Content Not Showing

```
# Check draft status
head -n 10 content/posts/my-post.md

# Include drafts in development
hugo server -D
```